List of changes in the program MCNelectron

Below is the list of changes for each version of MCNelectron that was available for download from http://web.vu.lt/ff/a.poskus/mcnelectron/, up to the current version (1.1.2).

1.0.1 (2014-06-29)**:**

1. In the current version, electron-positron pair production is simulated. However, a very simplistic model is used: both the electron and the positron continue in the direction of the incident photon, and positron energy distribution is assumed to be uniform.

2. In the current version, positron annihilation is simulated. Each annihilation event occurs when positron energy drops below the low-energy cutoff value.

3. In the current version, the source can emit positrons ("PART E+").

4. Fixed a bug in the procedure that is used to determine if a particle has escaped from the layer.

5. The current version implements a more realistic method of sampling of angular deflections of the two electrons existing after an electroionizatoin event. Their deflection angles are calculated from conservation of momentum.

6. The search for energy values in cross section tables has been made faster (this has been achieved by keeping in memory the current position in each table and using it as the starting position for the next search). As a result, the total simulation time has decreased by 20 % – 30 %.

1.0.2 (2014-07-01)**:**

1. In the current version, simulation of incoherent scattering at low energies (when atomic binding effects become important) is more realistic. The changes only affect the recoil electron. First, the atomic subshell is randomly selected on the basis of the number of electrons in each subshell. The energy of the ejected electron is obtained by subtracting the binding energy of the selected subshell from the energy loss of the photon. If the resulting energy is negative, then the electron is not ejected from the atom. Otherwise, the direction of its motion is the same as in the case of a free electron (only the energy is reduced as explained above). In previous versions, when tracking of particle position and direction was turned on ("TRACKPOS 1"), the effect of binding on the recoil electron energy was ignored (only the angular deflection included the effects of binding via the form factor).

2. In the output file, a new row "backscatter" has been added under "photon loss" and "electron loss". It contains the number and energy of particles that escaped from the layer through the surface that is exposed to the incident beam. The number and energy of photons or electrons that escaped from the other surface can be obtained by subtracting the values in the row "backscatter" from the corresponding values in the row "escape".

3. Fixed a bug that caused absence of angular deflection after coherent scattering of a photon.

1.0.3 (2014-07-04)**:**

1. The current version simulates Compton fluorescence. In the output file, the number and energy of the Compton fluorescence photons is shown separately from other fluorescence photons. However, the number and energy of electrons emitted during atomic relaxation after an incoherent scattering event is grouped with Auger electrons emitted after photoelectric absorption (similarly to MCNP6 output).

2. Fixed a bug that caused incorrect number of pair production events in the output file.

3. Added the keywords COH_THR, E_UNIT and E_CONTIGUOUS to the input file (see the updated README file for more information).

1.0.4 (2014-07-13)**:**

1. Added more information to the README file.

2. Added a subfolder "Test" with six MCNelectron input files and six MCNP6 input files corresponding to identical simulation conditions, together with respective output files. Additional information about those files and instructions on running the tests are in the file "Test_Test_info_.pdf".

1.0.5 (2014-07-19)**:**

1. Added the keyword NSTREAMS to the input file. It is used to specify the number of independent streams of random numbers. For optimum performance, NSTREAMS should be a multiple of the number of threads (see below). The maximum allowed number of streams is 256.

2. Added multithreading capability. The number of threads is specified on the command line in the same way as for MCNP, i.e., as "tasks <number_of_threads>". If there is no keyword "tasks" on the command line, then MCNelectron will run in single-thread mode. On multi-processor (or multi-core) systems, multithreading can significantly reduce simulation time. The number of threads can be assigned any value that does not exceed the number of streams of random numbers (see above). However, there is no point in using a number of threads that is greater than the number of logical processors in the system (a further increase of the number of threads will not cause a decrease of the simulation time). The final results of the simulation do not depend on the number of threads that was used during the simulation.

3. Updated the README file.

2014-07-26**:**

4. Updated the batch file "W-value\Make_W-tables.bat", which is used to build the final tables of W-values. The previous version of that file worked correctly with MCNelectron v1.0.4 output files, but it did not work with MCNelectron v1.0.5 output files. The updated version of "Make_W-tables.bat" works correctly both with v1.0.4 and with v1.0.5 output files.

2014-08-13**:**

5. Fixed a bug that could in extremely rare cases cause a crash after a positron annihilation event (probability of such a crash was less than 0.01 %).

1.0.6 (2014-08-23)**:**

1. Log-log interpolation of photon cross sections and form factors has been made faster. This has been achieved by calculating their natural logarithms before the simulation and then using those logarithms for linear-linear interpolation with natural logarithm of photon energy (or of the change of photon momentum) as input. As a result, the simulation time in problems where most of the time is spent for sampling photon interactions (e.g., a beam of photons incident on a thin layer) has become less than a half of the simulation time with MCNelectron v1.0.5.

2. When sampling the distance to collision, expression "1 - pRand->random()" has been replaced by "pRand->random()". Although statistically the former expression is correct, too (because it also corresponds to a uniform distribution), it involves one extra operation (subtraction), which is not necessary. This replacement has been done in two places in the code (one for photons and one for electrons). Because of this change, simulation results when some of the particles can escape from the layer are not exactly identical to the results obtained with the previous versions, although they are identical statistically.

3. Added an optional command-line keyword "output_cs". When that keyword is used on the command line, files with values of excitation and electroionization cross sections are created before starting the

simulation. In such a case, MCNelectron creates a folder "Cross sections" in the current folder, with subfolders for each of the chemical elements that are present in the target material. Each of those subfolders contains a text file with values of the total excitation cross section, as well as text files with electroionization cross sections for each subshell. Each of those files contains two columns of numbers: energy values (eV) and corresponding cross section values (in barns).

1.0.7 (2014-08-26)**:**

1. Interpolation of cross sections has been made faster by calculating the slope coefficients for each energy sub-interval before the simulation and using their stored values for interpolation (in previous versions of MCNelectron, the slope coefficients were recalculated during each step of the simulation). In addition, sampling of energy transfer during electroionization and bremsstrahlung, as well as sampling of angular deflection during elastic scattering of electrons has been made faster using a similar approach. As a result of those changes, the simulation time has decreased by 10 % to 20 % in comparison with MCNelectron v1.0.6.

2014-08-28**:**

2. Fixed a bug that appeared after the most recent change (described in previous paragraph). That bug caused incorrect simulation results when alternative cross sections were used, because then the corresponding slope coefficients were not recalculated (instead, their values corresponding to the original cross sections were used). After fixing that bug, the slope coefficients are recalculated using the updated set of cross sections.

3. Merged source files "thread.cpp", "thread.h" and "MCNelectron.h" with "MCNelectron.cpp". Consequently, the total number of source files decreased from 6 to 3.

1.0.8 (2014-08-30)**:**

1. In the current version of MCNelectron, the values of cross sections, form factors and other quantities that only depend on electron or photon energy are only recalculated when the particle energy changes. As a result of this modification, the simulation time in problems where particle energy changes relatively infrequently (i.e., when most collisions are without energy change, or when the source emits monoenergetic particles and the average number of energy-loss collisions per source particle is much less than 1) has decreased by 10 % to 40 % in comparison with MCNelectron v1.0.7. In "W-value problems", i.e., when both elastic scattering of electrons and coherent scattering of photons are switched off and the target is infinite in all directions ("TRACKPOS 0"), the computational overhead associated with this modification causes an increase of the simulation time by up to 10 %.

2014-09-06**:**

2. The runtime components of Visual C++ libraries (i.e., the "Microsoft Visual C++ Redistributable") are no longer required to be installed on a computer in order to run MCNelectron (before this fix, absence of the mentioned files caused the error message "The program can't start because MSVCR100.dll is missing from your computer" after attempting to run MCNelectron).

2014-09-12**:**

3. Fixed a bug that caused an error message about incorrect format of the ENDF file when the target material contained elements with atomic number greater than 86 (i.e., the elements that are after radon in the periodic table).

2014-09-26**:**

4. Fixed a bug that caused too small energies of Compton recoil electrons when tracking of secondary photons was turned off ("TRACK_P 0").

2014-09-27**:**

5. Fixed a bug that caused the failure to start the file "W-value\MakeTable.exe", if the runtime components of Visual C++ libraries (i.e., the "Microsoft Visual C++ Redistributable") are not installed on the computer.

1.0.9 (2014-10-02)**:**

1. Simulation of several physical effects has been added:

   (a) non-isotropic angular distribution of bremsstrahlung photons,
   (b) partial suppression of positron bremsstrahlung in comparison with electron bremsstrahlung,
   (c) non-isotropic angular distribution of photoelectrons,
   (d) non-uniform distribution of positron and electron energies during pair production,
   (e) angular distribution of electrons and positrons created in pair production events,
   (f) Doppler broadening of energy distribution of incoherently scattered photons.

   For each of those effects, there is a keyword in the input file, which allows "turning on" or "turning off" simulation of the effect. By default, all those effects are simulated. If any of those effects is "turned off", then it will be replaced by a simplified model (e.g., isotropic bremsstrahlung) or neglected completely, as in the previous version of MCNelectron. For some of those effects, there are other keywords controlling some details of the simulation (see the updated README file for more information). Simulation of bremsstrahlung angular distribution and Doppler broadening requires additional data files, which are included in the MCNelectron distribution package: the file "Data\Brems_angular_prob_dens.dat" with angular probability densities of bremsstrahlung (calculated by integrating analytically over the triply differential cross-sections derived by Bethe and Heitler) and the file "Data\ComptonProfiles.dat" with Compton profiles. The data in the file "Brems_angular_prob_dens.dat" is in binary format (it is described in the updated README file). However, there is an executable "Data\ExtractBremsAngularDistr.exe", which extracts ASCII data from the binary data and creates a subfolder with human-readable files. Those text files can be used as input data for the simulation, too (instead of the mentioned binary file).

2. The current version of MCNelectron can calculate a pulse-height tally (similar to the MCNP "F8" tally).

3. Added the keyword TRACK_X to the input file (it controls tracking of X-ray photons emitted due to atomic relaxation).

4. The energy of the Compton recoil electron is calculated simply as the difference of the incident and scattered photon energies (i.e., the recoil electron energy is no longer reduced additionally by subtracting the binding energy of the subshell from which the electron was ejected).

5. When the keyword "output_cs" is specified on the command line, a file with values of the total ionization cross section is created (in addition to the files with ionization cross sections for individual subshells).

6. If the ENDF data folder name, which is specified in the input file after the keyword DIR, does not contain the full path (i.e., if it does not start with the backslash '\' and does not contain the colon ':'), then it is now assumed to be a subfolder of the folder where the MCNelectron executable is (rather than a subfolder of the current folder). The same applies to the keyword "BREMS_ANGULAR_DISTR_DIR", which is used to specify location of text files with bremsstrahlung angular distribution data, when the mentioned binary file "Data\Brems_angular_prob_dens.dat" is not used.

7. Updated the README file.

2014-10-03**:**

8. Multiple pulse-height tallies are now allowed.

9. If a value of absorbed energy exceeds the bin boundary of a pulse-height tally by 10^{-10} % of the bin width or less, then that energy value is treated as though it is equal to that boundary. In a special case when that boundary is the low-energy limit of the tally energy range, the source particle is not counted.

2014-10-05**:**

10. Added the keyword INCOH_IGNORE_BINDING_E to the input file (see the updated README file for information about that keyword).

11. Renamed the keyword “DOPPLER_BROADENING” to “INCOH_DOPPLER”, in order to be consistent with the naming convention of the other keywords that control simulation of physical effects (the first part of those keywords is derived from the name of an interaction type).

2014-10-30**:**

12. Added the capability to equalize the workload of different threads in multi-threading mode. When a thread finishes processing of all histories assigned to it, the program redistributes all unfinished random number streams among the threads, so that the number of active threads stays the same (in the previous versions, the threads were not restarted after termination). This is only possible when the number of unfinished random number streams is not less than the number of threads (otherwise, each stream of random numbers is assigned to a separate thread, so that the number of active threads is equal to the number of unfinished random number streams).

2014-11-14**:**

13. Fixed a bug that could in extremely rare cases cause an indeterminate energy value (“-1.#IND”) of a scattered photon when simulating Doppler energy broadening during incoherent scattering. Such an error was possible when the sampled value of the projection of the electron initial momentum to the direction of incident photon is very close to zero, resulting in calculation of the square root of a difference of two very close numbers. Although the exact value of that difference is always non-negative, the rounding errors can make it negative (however, since there are no known cases of this actually happening, it is possible that “granularity” of the random number generator implemented in MCNelectron would prevent this from occurring anyway).

2014-11-29**:**

14. Fixed a bug that prevented periodic redistribution of random number streams among the threads to equalize their workload (because of that bug, the mentioned redistribution was only done once per simulation).

15. Increased the maximum allowed number of random number streams from 256 to 2048.

2014-11-30**:**

16. Fixed a bug that sometimes caused incorrect value of the positron bremsstrahlung suppression factor $\Phi_{\mathrm{rad}}^{+}/\Phi_{\mathrm{rad}}^{-}$ (although inequality $0.02 \le \Phi_{\mathrm{rad}}^{+}/\Phi_{\mathrm{rad}}^{-} \le 1$ was still always satisfied, as it should). This bug only showed up when there was non-zero probability of pair production by photons and the target material consisted of more than one chemical element, or when the source emitted positrons.

2014-12-03**:**

17. Fixed a bug that could cause an infinite loop in the part of the code that simulates photon Doppler energy broadening during incoherent scattering. Such a situation occurred when the binding energy of the selected atomic subshell happened to be greater than the incident photon energy. This bug has been fixed by rejecting such subshells, i.e., by selecting only subshells with binding energy less than the incident photon energy.

2015-05-15**:**

18. Fixed a bug that prevented update of components of the electron momentum vector after a bremsstrahlung event, when energy of the emitted photon was below the photon cutoff energy (the absolute value of that vector and the electron energy were updated correctly). This bug could have a noticeable effect only when electron energy is comparable to the photon cutoff energy (1 eV by default) and when particle coordinates are tracked.

1.1.0 (2015-06-20)**:**

1. Plane-crossing tallies have been added, together with corresponding keywords ETALLY and PTALLY for electron and photon plane-crossing tallies, respectively.

2. Removed restrictions on the angle of incidence and on the position of the starting point of the incident beam (it may be inside the layer).

3. Added an option to use inner-shell electron impact ionization cross sections calculated using the distorted-wave Born approximation (DWBA). The latter cross sections are the same ones that are used by the PENELOPE code system. They are calculated using the code by D. Bote, F. Salvat, A. Jablonski, and C. J. Powell, which was published in 2009. That code has been translated from FORTRAN into C using f2c.exe and incorporated into MCNelectron (it is contained in the file "IonCS.cpp").

4. Added CUDA support, together with 18 new keywords ending with "_CUDA" (15 of them may be used both in the input file and on the command line; the remaining 3 keywords are command-line only). As a result, Nvidia graphics processing units (GPU) can now be used for simulations in parallel with the CPU. There are now two variants of MCNelectron: the CPU-only one (executable file name MCNelectron.exe) and the one with CUDA support (executable file name MCNelectron_CUDA.exe). The file MCNelectron_CUDA.exe can perform GPU-only, CPU-only or hybrid CPU/GPU computations (in CPU-only mode, MCNelectron_CUDA.exe produces the same results as MCNelectron.exe).

5. Changed the random number generator. The new random number generator is the original Fibonacci series generator published by George Marsaglia and Arif Zaman in 1987. In addition, simulations using CUDA devices may optionally use the XORWOW random number generator from the cuRAND library (the latter generator can not be used on the CPU). The CPU random number generator was changed because the previous generator, which was a derivative of the one by G. Marsaglia and A. Zaman, was less suited for generation of a large number (≈30 000) of independent random number streams.

6. Added an option to apply a diffusion model of elastic scattering at extremely low electron energies, when the elastic scattering cross section is much larger than cross sections of other types of interactions. When the diffusion model is applied, the program first generates the distance to the next inelastic interaction event (i.e., the path traveled to that event, not the displacement), and samples the electron coordinates corresponding to that event from a Gaussian distribution with a root-mean-square displacement proportional to the square root of the generated path. This reduces the "electron trapping" effect (which could otherwise slow the simulation significantly) at the expense of fidelity of the simulated electron trajectory when its energy is only a few electronvolts above the ionization threshold of the material, or when it is below the ionization threshold. The transition to the diffusion model occurs at a user-specified value of the total-to-inelastic cross section ratio. The mentioned threshold value of that ratio is specified by the keyword "DIFFUSION_TOTAL_TO_INELASTIC_RATIO_THR". The allowed values of that threshold are from 10 to 10^{18}, and the default value is 100. The diffusion model of elastic scattering is never applied if the distance from the electron to the nearest surface of the layer is less than the mean free path times the square root of this parameter. In order to "turn off" application of the diffusion model, this parameter must be set to any sufficiently large value (e.g., to the maximum possible value 1e18).

7. The above-mentioned Gaussian distribution is only exact when the electron is at a depth d large enough to make the probability that the electron escapes from the layer before the next inelastic collision occurs practically zero. Consequently, the diffusion model of elastic scattering may introduce significant errors when estimating the moment and position of electron's escape from the layer if the electron is close to the surface. In order to decrease those errors, a single diffusion step mentioned above is replaced by a sufficiently large number of shorter diffusion steps, which are less than the root-mean-square displacement and also less than d (but still greater than the mean free path), if d is less than a user-specified number of r.m.s. displacements (the latter r.m.s. displacement is estimated as described above). This number is specified by the keyword "DIFFUSION_DIST_TO_RMSPATH_RATIO_THR". The allowed values of that number are from 2 to 10, and the default value is 5.

8. After an electron impact ionization event, the banked electron is the higher-energy ("primary") one (rather than the "knock-on" electron as in previous versions of MCNelectron). As a result, the number of banked electrons has been reduced significantly in comparison with previous versions (usually by an order of magnitude). An exception is a simulation on a CUDA device using the option "heap_CUDA 2": in this case the banked electron is the lower-energy one (i.e., the knock-on electron).

9. Replaced the keyword "INCOH_IGNORE_BINDING_E" by "INCOH_SUBTRACT_BINDING_E".

10. Updated the user manual.

2015-06-24**:**

11. Added the subfolder "X-rays" with 99 MCNelectron_CUDA input files and the corresponding output files. Each of them is used to simulate characteristic X-ray emission from a thick target composed of one of those elements: C, Al, Si, Ti, Fe, Cu, Zn, Ge, Zr, Sn, Sm, Ta, W, Pt, Au and Pb, bombarded by monoenergetic electrons with energy from 5 keV to 30 keV. Information about those files is provided in Section 8 of the user manual and in the file "_X-rays_info_.pdf", which is in the same subfolder.

2015-06-25**:**

12. Fixed a bug that caused the seed of the XORWOW random number generator from the cuRAND library to be always equal to the seed of the Fibonacci series random number generator, which is used on the CPU.

13. Fixed a bug that caused a CUDA error "invalid argument" after pressing any of the keys "+", "-", "[" or "]" during simulation using options "emFirst_CUDA 1" and "emLast_CUDA 1" on a computer with two or more CUDA devices (i.e., when the CUDA device No. 1 was being emulated by the CPU, while the CUDA device No. 2 was not being emulated).

2015-06-27**:**

14. The inner-shell electron impact ionization cross sections calculated using the DWBA approximation are now stored in the file "Data\IonDWBA.dat" in binary format, instead of being recalculated at the start of each simulation. Accordingly, the code for calculating those cross sections has been removed from MCNelectron.

15. Fixed a bug that caused incorrect values of inner-shell positron impact ionization cross sections when the option "ION_DWBA 1" was specified in the input file.

2015-07-03:

16. Added the file "randomlib.c" with the original code of the Fibonacci series random number generator to the distribution package; inserted a comment line with a reference to that file at the beginning of files "RandGen.cpp", "RandGen.cu" and "RandGen_CUDA.cu".

2015-07-15:

17. The error messages displayed after detecting an invalid command-line parameter have been made more informative.

2015-08-21:

18. Fixed a bug in the CUDA version of MCNelectron that sometimes caused overflow of the banked particles buffer when the option "heap_CUDA 2" was used. Now that overflow is prevented by controlling which of the two electrons existing after an impact ionization event is banked: when the banked particles buffer is less than 80 % full, the banked electron is the lower-energy one (i.e., the "secondary" or "knock-on" electron), and if the banked particles buffer is more than 80 % full, then the banked electron is the higher-energy one (i.e., the "primary" electron). In the case "heap_CUDA 0" or "heap_CUDA 1", the banked electron is always the primary one (as before).

1.1.1 (2015-10-02)**:**

1. Added an option to use elastic scattering cross sections obtained from relativistic (Dirac) partial-wave calculations. Those cross sections were calculated using the code ELSEPA by F. Salvat, A. Jablonski and C. J. Powell, which was published in 2005. These are the same cross sections that are used by the PENELOPE-2014 code system. Their values are stored in files "Data\ElasticDPW_totalCS.dat" and "Data\ElasticDPW_angularDistr.dat" in binary format. The latter file contains angular distributions, which were calculated for electron energies from 10 eV to 1 GeV on a logarithmic scale (the increment of the base-10 logarithm of energy is equal to 0.1). Usage of those cross sections is controlled by the switch "ELASTIC_DPW". By default, that switch is "on", i.e., those cross sections are used instead of the ENDF/B elastic scattering cross sections.

2. The default state of the switch “ION_DWBA” is “on”, i.e., 1 (previously, its default state was “off”, i.e., 0). Now, in order to avoid replacement of available ENDF/B data by third-party data, it is necessary to specify the following three options in the input file: “ION_DWBA 0”, “ELASTIC_DPW 0”, “BREMS_POSITRON_CORRECTION 0” (the latter option has an effect only when pair production is possible, or when the source emits positrons).

3. Added an option to reduce the size of the tables with values of the angular CDF of elastic scattering by removing some points from them (see the updated user manual for more details).

4. Added an option to speed up the process of searching for values in tabular data by “mapping” the tables to “index arrays” corresponding to sets of equidistant values (see the updated user manual for more details).

5. Added a range type “THETA”, corresponding to the angle between the particle’s momentum vector and the normal vector of a “tallying plane” (see the updated user manual for more details).

6. Increased the maximum number of random-number streams from 31329 to $2^{16} = 65536$. However, the maximum number of unique random number streams that can be generated by the Fibonacci series generator using a given seed (specified after the keyword “SEED”) is still equal to 31329. Consequently, the random number streams with sequence numbers from 31330 to $2 \cdot 31329 = 62658$ will be generated using the next seed (i.e., the seed that exceeds the specified seed by 1), and the streams with sequence numbers from 62659 to 65536 will be generated using the seed that exceeds the specified seed by 2 (if the seed value obtained by adding 1 or 2 is greater than 30081, then it is reduced by 30082).

7. Added four CUDA-specific keywords “use_CUDA”, “skip_CUDA”, “r_CUDA” and “em_CUDA”, which allow to use lists of CUDA devices, as well as to assign device workload ratios for individual CUDA devices (see the updated user manual for more details).

8. Added a CUDA-specific keyword “bankModeThr_CUDA”, which controls the “bank mode” when option “heap_CUDA 2” is used. The term “bank mode” refers to the choice of an electron that has to be banked after an impact ionization event: the banked electron may be either the lower-energy one (the so-called “secondary” or “knock-on” electron), or the higher-energy one (the “primary” electron). Normally, when option “heap_CUDA 2” is used, the knock-on electron is the one that is banked. However, this may cause an overflow of the banked particles buffer in the case of high energies of source particles and thick targets. Consequently, the bank mode is changed when the filling fraction of the banked particles buffer exceeds a certain value. That “threshold” value of the filling fraction is specified after this keyword. Its default value is 0.8, i.e., the bank mode will be changed when the banked particles buffer is more than 80 % full. In the case “heap_CUDA 0” or “heap_CUDA 1”, this keyword is ignored (in those cases, the primary electron is always the one that is banked).

9. Added four command-line keywords “NSTREAMS”, “SEED”, “N” and “E”. The first three ones are equivalent to corresponding keywords specified in the input file and allow overriding the number of random-number streams, the seed of the Fibonacci series random number generator and the number of source particles specified in the input file, respectively. The keyword “E” makes it possible to replace the definition of the source energy spectrum specified in the input file by a source of monoenergetic particles.

10. Added three command-line keywords “in”, “out” and “alt” that must precede the names of the input file, the output file and the file with alternative cross sections information, respectively (previously, the file names were entered without any keywords).

11. Moved subfolders “W-value” and “X-rays” into a new subfolder “Simulations”. Added a subfolder “Simulations\Backscattering” with results of simulations of electron backscattering from a thick layer of gold at incident electron energies from 200 eV to 15 MeV, obtained with MCNelectron_CUDA using the option “ELASTIC_DPW 1”.

12. MCNelectron and MCNelectron_CUDA are now distributed as WinRAR self-extracting archives. The names of those files are "MCNelectron_install.exe" and "MCNelectron_CUDA_install.exe", respectively. They are contained in WinZip archives "MCNelectron_install.zip" and "MCNelectron_CUDA_install.zip", respectively (they replace the files "MCNelectron.zip" and "MCNelectron_CUDA.zip", which were used previously).

13. Updated the user manual.

2015-10-10:

14. Fixed a bug that caused inconsistent sign of a bin count increment (±1) when a range type "MU" was used in the specification of a two-dimensional plane-crossing tally (before fixing this bug, the mentioned sign depended on the direction of crossing, whereas for this range type the sign should be always positive).

15. Fixed a bug in the CUDA version of MCNelectron that caused inconsistent sign of a bin count increment (±1) when a range type "THETA" was used in the specification of a two-dimensional plane-crossing tally (before fixing this bug, the mentioned sign depended on the direction of crossing, whereas for this range type the sign should be always positive). This bug affected only the 2D tallies calculated on CUDA devices (but not the tallies calculated on the CPU).

2015-10-14:

16. Fixed incorrect interpretation of a negative value of the input parameter ELASTIC_PDF_ERROR in the CPU-only version of MCNelectron. Before fixing this bug, a negative value of ELASTIC_PDF_ERROR was always interpreted by the CPU-only version of MCNelectron as a zero value (i.e., no values were removed from tables with values of elastic scattering angular PDF), whereas the correct way is to ignore the mentioned sign when the option "ELASTIC_DPW 1" is used (as explained in the user manual). ***Note***: This bug was absent in the CUDA version of MCNelectron.

1.1.2 (2015-11-10)**:**

1. Added an option to use interaction forcing (see Section 4.5 in the updated user manual).

2. The files with plane-crossing and pulse-height tally data now include relative standard deviations for each bin (in addition to the particle counts).

3. Added an option to terminate the simulation when the maximum, minimum or average relative standard deviation, calculated over all bins of one or more plane-crossing or pulse-height tallies ("control tallies"), becomes less than the "target error" specified by the user (see Section 4.4 in the updated user manual).

4. The simulations of electron-induced characteristic X-ray emission (folder "Simulations\X-rays" of the distribution packages) have been re-done using interaction forcing and "control tallies".

5. Updated the user manual.

2015-11-11:

6. Fixed a bug in MCNelectron_CUDA that could cause "loss" of one source particle during a GPU-only simulation using two or more CUDA devices with automatic workload balancing (this bug was caused by rounding errors when converting a floating-point number to an integer number).

2015-11-12:

7. Fixed a bug that caused exclusion of forced interactions from the sampled interaction types during analog tracking of particles with weights below the cutoff value. This bug could affect the simulation results only when at least one of parameters FORCING_WT_MIN or CUT_WT was not equal to zero (which is the default value of those parameters).

2015-12-02:

8. Fixed a bug in MCNelectron_CUDA that caused failure to terminate a CPU-only simulation after the minimum, maximum or average relative standard deviation of a “control tally” became less than the “target error”.

9. Fixed a bug in MCNelectron_CUDA that caused incorrect value of the current minimum, maximum or average relative standard deviation when checking for a termination condition in a two-dimensional “control tally”.

10. Corrected the language in several error messages.

2015-12-07:

11. Fixed a bug that could cause a crash or incorrect results when the specification of the “target error” of a “control tally” was not the last entry of the line with the tally definition in the input file.